# CWE Detail – CWE-1332

## Description

The device is missing or incorrectly implements circuitry or sensors that detect and mitigate the skipping of security-critical CPU instructions when they occur.

## Extended Description

The operating conditions of hardware may change
 in ways that cause unexpected behavior to occur,
 including the skipping of security-critical CPU
 instructions. Generally, this can occur due to
 electrical disturbances or when the device operates
 outside of its expected conditions. In practice, application code may contain
 conditional branches that are security-sensitive (e.g.,
 accepting or rejecting a user-provided password). These
 conditional branches are typically implemented by a
 single conditional branch instruction in the program
 binary which, if skipped, may lead to effectively
 flipping the branch condition - i.e., causing the wrong
 security-sensitive branch to be taken. This affects
 processes such as firmware authentication, password
 verification, and other security-sensitive decision
 points. Attackers can use fault injection techniques to
 alter the operating conditions of hardware so that
 security-critical instructions are skipped more
 frequently or more reliably than they would in a
 "natural" setting.

## Threat-Mapped Scoring

Score: 3.0

Priority: P2 - Serious (High)

## Observed Examples (CVEs)

**•** CVE-2019-15894: fault injection attack bypasses the verification mode, potentially allowing arbitrary code execution.

## Related Attack Patterns (CAPEC)

* CAPEC-624
* CAPEC-625

## Modes of Introduction

**•** Architecture and Design: Failure to design appropriate countermeasures to common fault injection techniques can manifest this weakness.

**•** Implementation: This weakness can arise if the hardware design incorrectly implements countermeasures to prevent fault injection.

## Common Consequences

**•** Impact: Bypass Protection Mechanism, Alter Execution Logic, Unexpected State — Notes: Depending on the context, instruction skipping can
 have a broad range of consequences related to the
 generic bypassing of security critical code.

## Potential Mitigations

**•** Architecture and Design: Design strategies for ensuring safe failure if
 inputs, such as Vcc, are modified out of acceptable
 ranges. (Effectiveness: N/A)

**•** Architecture and Design: Design strategies for ensuring safe behavior if
 instructions attempt to be skipped. (Effectiveness: N/A)

**•** Architecture and Design: Identify mission critical secrets that should
 be wiped if faulting is detected, and design a
 mechanism to do the deletion. (Effectiveness: N/A)

**•** Implementation: Add redundancy by performing an operation
 multiple times, either in space or time, and perform
 majority voting. Additionally, make conditional
 instruction timing unpredictable. (Effectiveness: N/A)

**•** Implementation: Use redundant operations or canaries to
 detect and respond to faults. (Effectiveness: N/A)

**•** Implementation: Ensure that fault mitigations are strong enough
 in practice. For example, a low power detection
 mechanism that takes 50 clock cycles to trigger at lower
 voltages may be an insufficient security mechanism if
 the instruction counter has already progressed with no
 other CPU activity occurring. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** There are several ways this weakness could be fixed.